Checking properties after shallow parsing

Luísa Coheur[1] and Nuno J. Mamede[2]

[1] L^{2}F – INESC-ID Lisboa/IST/GRIL
[2] L^{2}F – INESC-ID Lisboa/IST
Spoken Language Systems Lab
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{Luisa.Coheur,Nuno.Mamede}@inesc-id.pt

Resumo Tira-Teimas is a program that verifies if a shallow parsed text satisfies a set of properties from the 5P paradigm. Tira-Teimas is implemented in XSLT and it is automatically generated by another program called Tira-Teimas Translator, which is also implemented in XSLT.

1 Introduction

The 5P paradigm [1,2,3] uses a set of properties to describe natural language syntax and Hagège in [3] made a precise description of Portuguese nominal phrases using these 5P properties. Also, she developed a shallow parser prototype called AF [3,4], and enriched the structures used by AF with the information expressed within these properties. Nevertheless, AF's information structures are less expressive than 5P properties. Therefore, it is not sure that the sequences identified by AF verify the whole set of 5P properties.

Tira-Teimas was then built in order to check if there is any inconsistency between the information from 5P properties and AF's results. Therefore, Tira-Teimas is a program that verifies if a shallow parsed text satisfies or not a set of properties as it checks if each sequence identified by AF satisfies (or not) the 5P properties.

The following chapter briefly describes 5P properties, AF's structures, and emphasize the gap between these formalisms. Then, in Section 3 it is explained how Tira-Teimas was built. An auxiliary tool, called Tira-Teimas translator is introduced in Section 4. Section 5 shows some results and we conclude the document with the usual conclusions and future work.

2 5P properties, blocks and leaves

2.1 5P properties: a brief overview

In the following we briefly describe 5P properties (see [1,2] or [3] for extra details). This presentation is illustrated with examples from [3], where a detailed

description of the syntax of Portuguese nominal phrases (excluding coordination) can be found.

5P properties are divided into three sets: existence properties (vocabulary, uniqueness, nucleus, exigency and exclusion), linearity and arrow properties.

Vocabulary (V) This property allows to declare the elements (models or categories) occuring within a model. As an example

$$\mathrm{V}_{Nn} = \{\mathsf{n}, \mathsf{adj}, \mathsf{card\_p}, ...\}$$

indicates that inside a nuclear nominal phrase (Nn), we can have elements labeled by the elements front V_{Nn} or labeled by elements that represent a category subsumed by an element from V_{Nn}. In this particular example, the category labeled n(oun) subsumes the category for common nouns (singular, plural, ...) as well as proper nouns (of type 1, 2, 3 and 4). Therefore, elements from these categories can occur inside an Nn. The same happens with the categories subsumed by adj(ective). The category labelled card_p does not subsume any other category, meaning that an element labeled card_p can appear inside an Nn.

Uniqueness (Un) With this property we are able to identify the elements that cannot occur more than once in a certain model. As all example

$$\mathrm{Un}_N = \{\mathrm{Nn}\}$$

indicates that inside a nominal phrase (N), there is, at most, one nuclear nominal phrase Nn.

Nucleus (Nu) This property declares which are the elements that can be the nucleus of the phrase[3]. As so,

$$\mathrm{Nu}_N = \{\mathsf{Nn}\}$$

means that only a nominal nuclear phrase can be the nucleus of a nominal phrase.

Exigency (⇒) Exigency properties allow to declare that a certain element occurs is a model only if another element also occurs in it. As an example

$$\{\mathsf{mesmo}\} \Rightarrow_{Nn} \{\mathsf{dem}\} \mid \{\mathsf{artd}\},$$

means that inside a nuclear nominal phrase, if there is an element labeled mesmo, then an element labeled dem(onstrative) or artd (definite article) must also occur

[3] In each model there is one and only one nucleus.

(ex: *o mesmo*, *este mesmo carro*, *esse mesmo rapaz*)[4].

We can have the following extra-notation for an exigency property:

$\mathsf{a} \Rightarrow_X$ NIL

meaning that a must be alone in a model labelled X (it excludes everything).

Exclusion ($\not\Rightarrow$) These properties permit to declare that an element excludes another element within a model. For example, with

{ord} $\not\Rightarrow_{Nn}$ {nc2}

we have that an ord(inal) and a common noun of type 2 (nc2) exclude mutually inside a Nn (ex: *A terceira água*)[5].

Linearity ($<$) These properties are responsible to declare linearity relations between the elements of a model. Consider

det $<_{Nn}$ poss

It indicates that if a det(erminer) and poss(essive) occur in Nn, then the det precedes the poss (ex: *o meu livro*)[6].
We say that a model verifies these properties if it does not disrespects any of them. As an example, *meu o livro*[7] does not satisfies the presented properties because *meu* (poss) precedes the det *o*, desrespecting the linearity property.

2.2 Blocks and leaves

Hagège developed a surface analyser AF (the leaves algorithm), that receives, as input, two data structures called leaves and blocks. Briefly, each model has a block associated and each category has a leaf. Both leaves or blocks indicate:

- which is the model where they are in (the upper model);
- if they can always(1)/never(O)/sometimes(2) start the upper model;
- if they can always(l)/never(O)/sometimes(2) end the upper model;
- which are the elements that can appear after them in the upper model.

That is, each leaf and each block have the following structures, respectively:

---

[4] Respectively, *the same*, *this same car*, *the same boy.*
[5] *The third water.*
[6] *My book.*
[7] **my the book.*

leaf(_, category, upper model, start, end, next elements)

block(current model, upper model, start, end, next elements)

where

start, end $\in \{0, 1, 2\}$.

2.3 The gap

The problem is that the blocks and the leaves do not have the same expressiveness of 5P properties. As an example, consider the set of sequences labelled X and described by the following properties (suppose that a, b and c are categories from the vocabulary, and to simplify, that they are unique and that they can all be the nucleus of a model):

(1) $\mathsf{a} \Rightarrow_X \mathsf{b}$
(2) $\mathsf{a} \not\Rightarrow_X \mathsf{c}$
(3) $\mathsf{a} <_X \mathsf{b}$

The following models (labeled X) respect these properties:

$(\mathsf{c})_X$
$(\mathsf{b})_X$
$(\mathsf{a\ b})_X$
$(\mathsf{b\ c})_X$
$(\mathsf{c\ b})_X$

However, the following ones do not respect them:

$(\mathsf{a})_X$
$(\mathsf{b\ a})_X$
$(\mathsf{a\ c})_X$
$(\mathsf{a\ b\ c})_X$
...

In order to accept the first set of sequences, AF needs the following leaves:

leaf(_, a, X, 1, 0, [b]);
leaf(_, b, X, 2, 2, [c]);
leaf(_, c, X, 2, 2, [b]).

because

- if a precedes and exiges b, then a always starts and never ends a model;

– as nothing is said about the linearity between b and c, c can precede b or appear after it, meaning that b may start or end (or not) the model;
– *mutatis mutandis* for c.

Therefore, with these leaves, the following sequence that was not verifying the 5P properties

$(\mathsf{a}\ \mathsf{b}\ \mathsf{c})_X$

would be consider an X model by AF, because:

– a may start a model;
– after a a b can occur;
– c can follow a;
– c can end a model.

3 Building Tira-Teimas

3.1 Choosing XSLT

Susana [5] is, in rough terms, a new implementation of AF. As Susana's output is in XML [6], we decided to use XSLT (Extensible Stylesheet transformation) [7] in order to implement Tira-Teimas. XSLT is a tree-oriented transformation language, that uses stylesheets containing template rules and transforms XML into XML, HTML, txt, etc. As so, Tira-Teimas is nothing more than a stylesheet containing the 5P properties. It picks a shallow parsed text in XML, looks inside the detected sequences, and returns another shallow parsed text in XML, where the sequences not respecting 5P properties are identified and marked (see Fig. 1).

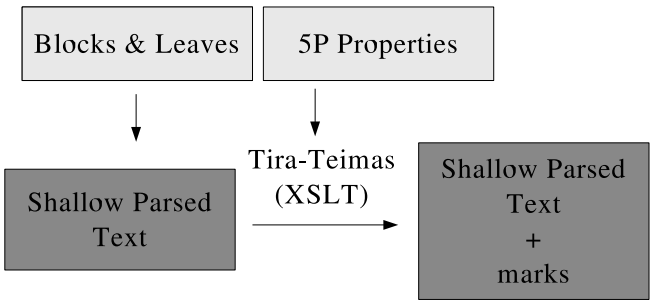

Figura 1. Tira-Teimas

3.2 Coding 5P properties in XSLT

In the following, we explain how 5P properties are coded in XSLT.

Exigency properties

Exigency properties have the following general syntax:

$$\text{Exig}_i\text{: } \{\mathsf{a}_1, ..., \mathsf{a}_n\} \Rightarrow_M \{\mathsf{b}_1, ..., \mathsf{b}_m\} \mid ... \mid \{\mathsf{c}_1, ..., \mathsf{c}_k\}$$

meaning that if the symbols a_1, ..., a_n, occur in a model labelled M, then

$$\{\mathsf{b}_1, ..., \mathsf{b}_m\} \text{ or ... or } \{\mathsf{c}_1, ..., \mathsf{c}_k\}$$

must also occur in that model[8].

Given this, suppose that a model labelled M is detected by the shallow parser. Then we can check if this property (Exig_i) is verified, by counting the number of occurrences of a, b and c in the model labelled M. That is:

Let count(x, X) be a function returning the number of occurrences of x in (the model) X. If

$$\begin{array}{l} \mathsf{count}(\mathsf{a}_1, \mathsf{M}) \neq 0 \text{ and ... and } \mathsf{count}(\mathsf{a}_n, \mathsf{M}) \neq 0 \\ \quad \text{and} \\ [(\mathsf{count}(\mathsf{b}_1, \mathsf{M}) = 0 \text{ or ... or } \mathsf{count}(\mathsf{b}_m, \mathsf{M}) = 0) \\ \quad \text{and ... and} \\ (\mathsf{count}(\mathsf{c}_1, \mathsf{M}) = 0 \text{ or ... or } \mathsf{count}(\mathsf{c}_k, \mathsf{M}) = 0)] \end{array}$$

then M does not satisfies Exig_i.

If the model M does not satisfies Exig, then it is marked with the identification of Exig_i.

As a predicate "count" is available in XSLT, mapping the previous formulas in XSLT is a trivial task.

Exigency properties – an example

As an example, consider an exigency property labelled Exig_{30}, defined as follows:

$$\text{Exig}_{30}\ \{\mathsf{a}, \mathsf{b}\} \Rightarrow_A \{\mathsf{c}\}$$

It will be coded in XSLT as it follows:

[8] These elements or others subsumed by them.

```
<xsl:template match='model[@name="A"]'>
<xsl:variable name="submodels" select="model"/>
<model>
<xsl:attribute name="name"><xsl:value-of select="@name"/>
</xsl:attribute>
<xml:if test="count($submodels[@name='a']!=0 and
              count($submodels[@name='b'])!=0 and
              ((count($submodels[@name='c'])=0))">
<exig>
   <xsl:attribute name="id">E30</xsl:attribute>
</exig>
<xsl:if>
```

This XSLT file runs over a shallow parsed text and the segments not respecting this property are identified and marked. For example, if after a surface analyisis we have the segment $(\mathsf{a}\ \mathsf{b})_A$, that is:

```
<model name="A">
   <model name="a"/>
   <model name="b"/>
</model>
```

the transformation operated by Tira-Teimas returns:

```
<model name="A" exig="30">
   <model name="a"/>
   <model name="b"/>
</model>
```

which indicates that property number 30 was not respected.

In conclusion, we codify 5P properties in a stylesheet that will transform a XML text in another XML text where the sequences not verifying the 5P properties are marked.

Other properties

A similar approach is applied to exclusion properties. Uniqueness can also be treated through element's counting as in the following.

Consider the property

$$\text{Un} = \{a_1, ..., a_n\}.$$

If

$\mathsf{count}(\mathsf{a}_1, \mathsf{M}) \leq 1$ and ... and $\mathsf{count}(\mathsf{a}_n, \mathsf{M}) \leq 1$

then the model M verifies this property. It should be noticed that a model with a coordination do not need to respect these properties.

4 Tira-Teimas Translator: an auxiliar tool

4.1 The problem

Writing 5P properties directly as a XSLT template is not an easy task. In fact, the complexity of the templates increases with the subsumption relations. The next example shows how difficult it is to write a basic 5P property as a XSLT template (in the following, det subsumes artd_s, artd_p, dem and arti_s).

The property

$\text{Exig}_3\ \{\mathsf{npr2}\} \Rightarrow_{Nn} \{\mathsf{det}\}$

originates the following template:

```
<xsl:template match='model[name="m_nn"]'>
<xsl:variable name="submodels' select="model"/>
<model>
<xsl:attribute name="name">
      <xsl:value-of select= "@name"/>
</xsl:attribute>

<xsl:if test="count($submodels[@name='npr2'])!=0 and (((
    count($submodels[@name='artd_s']) =0 and
    count($submodels[@name='artd_p']) =0 and
    count($submodels[@name='dem'])=0 and
    count($submodels[@name='arti_s'])=0 )))">
<exig>
   <xsl:attribute name="id">Exig3</xsl:attribute>
</exig>
</xsl:if>
```

4.2 Tira-Teimas Translator: generating Tira-Teimas

In order to simplify this situation, 5P properties are written in a friendlier XML format, and then an extra program – Tira-Teimas Translator (TTT) – maps these properties into XSLT. That is, TTT generates Tira-Teimas (see Fig. 2).

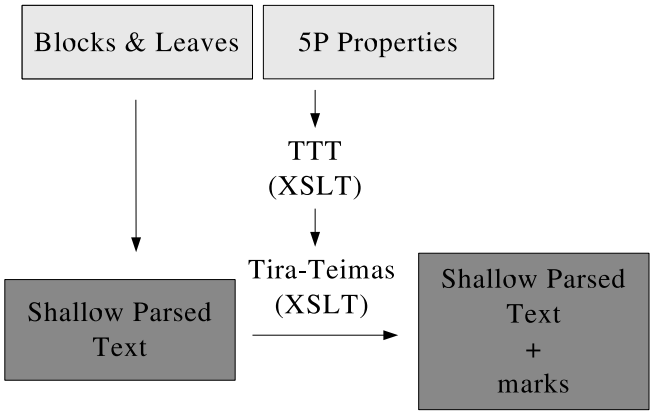

Figura 2. Tira-Teimas Translator

4.3 An example

In order to implement TTT we also used XSLT. In the following the property presented before is coded in the new XML format:

```
<exig m_label="m_nn" num="Exig3">
<left_exig><element id="npr2"/></left_exig>
<right_exig><element id="det"/></right_exig>
</exig>
```

TTT picks up the properties written in this format and returns Tira-Teimas.

5 Results

We took several (non treated) corpora, such as a tale, a thesis' chapter, a text about metereology and others and we parsed them with Susana. Also, we coded the set of 5P properties describing nominal phrases, presented in [3], in XML, in the friendlier format preciously discussed. Then, TTT run over the 5P properties in XML, generating Tira-Teimas. Finally, Tira-Teimas checked the shallow parsed corpora. In the following we show the most interesting situations we got.

Consider the exigency property number 15[9]:

Exig_{15}: adj_s $\Rightarrow$ det | algum | qualquer | ... | tanto

[9] algum, qualquer, ..., tanto are very particular category labels for the words *algum*, *qualquer*, ..., *tanto*, respectively.

Consider also the following two hypothesis obtained by Susana:

$(muito_{q3\_s}\ alto_{adj1\_s})_{Nn}$
$(((muito_{adv21_1})_{Advn}\ alto_{adj1\_s})_{An1})_{Nn}$

Tira-Teimas detected that the first hypotheses was not respecting property Exig_{15}, as none of the elements needed by adj1_s (subsumed by adj_s) are in the sequence. So, we are now able to cut the first hypotheses, solving an ambiguity situation.

Consider now the following property:

Exig_5: npr1 $\Rightarrow_{Nn}$ NIL

and Susana's result:

$(a_{art\_s}\ Dublin_{npr1})_{Nn}$
$(a_{prep}\ Dublin_{npr1})_{Pn}$

According to Tira-Teimas, the first hypotheses goes against property Exig_5, as inside an Nn, an npr1 should be alone. Once again, an ambiguity situation can be eliminated.

Now, take the exclusion property number 25:

Exclu_{25}: ppas $\not\Rightarrow$ adj3

Susana returned the following sequences:

$(um_{artd\_p}\ comprimido_{ppas}\ vermelho_{adj3\_p})_{Nn}$
$(um_{artd\_p}\ comprimido_{nc1\_s}\ vermelho_{adj3\_p})_{Nn}$

As within an Nn, adj3 excludes ppas, the first sequence is not possible.

To conclude, consider the property:

Exig_4: npr4 $\Rightarrow_{Nn}$ det | algum | qualquer | ... | card_p

and the following result:

$(Alfredo_{npr4})_{Nn}$ *teve...*

if this property is correct an npr4 (proper nouns for people's names), can not appear alone within an Nn. As this is a possible situation, we can conclude that this property is not correct.

6 Conclusions and further work

Tira-Teimas guarantees that models not respecting 5P properties are identified. As so, bad results can be eliminated. Also, bad properties can be eliminated.

Although the original motivation for Tira-Teimas was to check 5P properties, Tira-Teimas is not bounded to this application. In fact, it can also be used to find differences between what is syntactically correct and what is currently practiced.

In addition, Tira-Teimas could be applied to detect differences between the Portuguese from Portugal and Portuguese front Brazil. For example, the 5P properties from [3] describing Portuguese (from Portugal) nominal phrases could be applied to a Brazilian shallow parsed text.

Another possible application to Tira-Teimas would be to retrieve the 5P properties from a disambiguated shallow parsed text. All the possible 5P properties would be automatically generated and all the properties that the shallow parsed text disrespects would be possible restrictions over the language syntax.

Referências

1. Bès, G.G.: La phrase verbal noyau en français. In: in Recherches sur le français parlé, 15, Université de Provence, France (1999) 273–358
2. Bès, G.G., Hagège, C.: Properties in 5P (soon in the GRIL pages). (November, 2001)
3. Hagège, C.: Analyse Syntatic Automatique du Portugais. PhD thesis, Université Blaise Pascal, Clermont-Ferrand, France (2000)
4. Bès, G.G., Hagège, C., Coheur, L.: Des propriétés linguistiques à l'analyse d'une langue. In: VEXTAL, Venice, Italy (1999)
5. Batista, F., Mamede, N.: SuSAna: Módulo multifuncional da análise sintáctica de superfície. In Gonzalo, J., Peñas, A., Ferrández, A., eds.: Proc. Multilingual Information Access and Natural Language Processing Workshop (IBERAMIA 2002), Sevilla, Spain (2002) 29–37
6. World Wide Web Consortium (W3C): (Extensible Markup Language (XML)) See: www.w3.org/XML.
7. World Wide Web Consortium (W3C): (The Extensible Stylesheet Language (XSL)) See: www.w3.org/Style/XSL.